Zeinab Torabi
Somayeh Timarchi

SIGN DETECTION AND SIGNED INTEGER COMPARISON FOR THREE-MODULI SET $\{2^n \pm 1, 2^{n+k}\}$

Abstract *Comparison, division, and sign detection are considered to be complicated operations in a residue number system (RNS). A straightforward solution is to convert RNS numbers into binary formats and then perform complicated operations using conventional binary operators. If efficient circuits are provided for comparison, division, and sign detection, the application of RNS can be extended to those cases that include these operations.*
For RNS comparison in three-moduli set $\tau = \{2^n - 1, 2^{n+k}, 2^n + 1\}, (0 \leq k \leq n)$, we have found only one hardware realization. In this paper, an efficient RNS comparator is proposed for moduli set τ, which employs a sign-detection method and operates more efficiently than its counterparts. The proposed sign detector and comparator utilize dynamic range partitioning (DRP), which has been recently presented for unsigned RNS comparison. The delay and cost of the proposed comparator are lower than the previous works, which makes it appropriate for RNS applications with limited delay and cost.

Keywords computer arithmetic, residue number system, signed integer comparison, dynamic range partitioning

Citation Computer Science 22(3) 2021: 391–405

1. Introduction

A number X in a residue number system (RNS) with k moduli $\{m_1, m_2, \cdots, m_k\}$ is represented by set of residues $\{x_1, x_2, \cdots, x_k\}$, where $x_i = |X|_{m_i}$ denotes the remainder of integer division X/m_i. If the moduli are pair-wise prime, dynamic range M is maximized (i.e., $M = \Pi_{i=1}^{k} m_i$). In RNS, operations like addition, subtraction, and multiplication are performed in k parallel independent channels, which makes it a promising candidate for applications that use frequent add/multiply operations such as finite impulse response digital filters [4], data transmission [1], cryptography [18], and image processing [28]. Furthermore, digital signal processing (DSP) has employed RNS due to such properties as carry-free operations, parallelism, and modularity [2]. Recently, RNS has been used to achieve the energy-efficient hardware implementation of neural networks for inference computations [16, 17].

Since RNS is a non-weighted number system; comparison, division, sign, and overflow detection have difficulties. Difficult operations are often required for several nonlinear procedures, such as median and rank-order filtering [20]. Several RNS comparison [3, 5, 10, 12, 13, 20, 22–24, 27], sign-detection [9], and division [25] methods have been proposed over the past two decades. As regarding the role of comparison in other complicated operations such as division, sign, and overflow detection, it is expected that a cost-effective and high-speed implementation of the comparison will assist in improving other complicated operations. Moreover, the RNS comparators used in some applications such as video coding [19] and deep neural networks [17] lead to better results than a binary number system.

Besides popular moduli set $\{2^n \pm 1, 2^n\}$, several three-moduli sets with dynamic ranges of more than $3n$-bit have been reported in the relevant literature (along with their reverse converters) [6, 7, 14, 15]. Moduli set $\tau = \{2^n - 1, 2^{n+k}, 2^n + 1\}$ has efficient balanced arithmetic operations, with a $(3n + k)$-bit dynamic range and efficient reverse converter [8]. Besides the general methods for RNS comparison, there is only one comparator that is designed specifically for this moduli set [20]. By considering the efficiency of the dynamic range partitioning method for unsigned number comparison (as is experienced in several three-moduli sets), we are motivated to apply this method for sign detection and signed number comparison to moduli set τ. In this paper, we propose a signed number comparator for moduli set τ by employing the DRP method. With analytical evaluations, we show that the proposed DRP-based signed integer comparator outperforms other previous methods [8, 20]. The work of [8] only provides a reverse converter for moduli set τ, so we have augmented its converter with a normal binary comparator.

The rest of this paper is organized as follows. Section 2 reviews a number of efficient RNS comparison methods in the literature. In Sections 3 and 4, the proposed signed RNS comparator for moduli set τ are described (along with its implementation details). Section 5 provides a comparison of the proposed comparator with the existing schemes presented in [20] and [8]. Finally, we draw our conclusions based on the obtained results in Section 6.

2. Background material

Here, we first describe the representation of signed numbers in RNS and then review some previous RNS comparison schemes. The dynamic range of representing numbers in an arbitrary moduli set (i.e.,$[0, M)$) is partitioned into two nearly equal parts to provide negative numbers. $[0, [M/2])$ and $[[M/2], M)$ are considered to be positive and negative intervals, respectively, where the absolute value of negative number X is $M - X$. The sign of an RNS number X can be detected as follows:

$$\text{sign(X)}= \begin{cases} 0 & \text{if} \quad 0 \leq X < [M/2] \\ 1 & \text{if} \quad [M/2] \leq X < M. \end{cases}$$

Example: Consider moduli set $\{3, 4, 5\}$ with $M = 60$. $X = (1, 2, 3) = 58 > [60/2]$; therefore, X is a negative number. The absolute value of X equals $M - X = 60 - 58 = 2$; since X is negative, we have $X = (1, 2, 3) = -2$.

Several approaches have attempted to develop RNS comparison methods (mostly for unsigned numbers) [3,5,10,12,22–24,27]. Some other works compared signed numbers from a different perspective in which the dynamic range included both positive and negative numbers [13, 20].

The straightforward technique utilizes a reverse convertor to convert RNS operands to binary and then compares them with a normal comparator. Such convertors are usually based on Chinese remainder theorem (CRT) [21] or mixed-radix conversion (MRC) [21]. Subsequently, there are some other comparators [2, 12, 24, 27] that do not convert numbers completely and compare operands during the reverse-conversion process.

In [10], a method for non-modular operations in RNS by summation sets of floating-point numbers is proposed; however, this method is efficient only on modern massively parallel general-purpose computing platforms such as GPU-based systems.

The authors of [13] proposed a fast signed number comparator for three four-moduli sets based on the simple quantization method. Such a method exploits the unique number theoretic properties of the moduli.

Some other methods have proposed the diagonal function [3, 5] to compare RNS numbers. A diagonal number based on (1) is assigned to each number in the dynamic range. For comparing two operands X and Y, $D(X)$ and $D(Y)$ are computed; the result of comparison is then determined by comparing $D(X)$ and $D(Y)$. Like CRT, this method is based on modular operations in a large modulo SQ, where $SQ = \Sigma_{i=1}^{n}(M/m_i)$, and the values of μ_i satisfy $|\mu_i m_i|_{SQ} = 1$.

$$D(x) = |\mu_1 x_1 + \mu_2 x_2 + \cdots + \mu_n x_n|_{S_Q} \tag{1}$$

The works of [22, 23] compare unsigned RNS numbers via DRP in moduli sets $\{2^n, 2^n \pm 1\}$ and $\{2^n, 2^n - 1, 2^{n+1} - 1\}$. Partitioning the dynamic range of an arbitrary three-moduli set (such as $\{m_1, m_2, m_3\}$) produces m_1 partitions of a size of $m_2 \times m_3$ that are assigned to a primary integer identifier in $[0, m_1)$. Likewise, each partition is divided into m_2 partitions of a size of m_3, with a corresponding secondary

integer identifier in $[0, m_2)$. On the other hand, there are exactly one primary integer identifier and one secondary integer identifier for each operand regarding the RNS comparison. Therefore, the comparison operation can be reduced to comparing their primary and secondary integer identifiers (represented by $p_1(X)$ and $p_2(X)$) in (2) and (3), respectively, as well as the modulo-m_3 residues.
Equations (2) and (3) (which are borrowed from [22]) show how the DRP components of an operand $X = (x_1, x_2, x_3)$ can be derived from the corresponding RNS residues, where $x_{23} = |X|_{m_2 m_3}$, $M_1 = m_2 m_3$, and the multiplicative inverse of $|A^{-1}|_B$ satisfies $|A \times A^{-1}|_B = 1$.

$$p_2(X) = ||m_3^{-1}|_{m_2}(x_2 - x_3)|_{m_2}, \tag{2}$$

$$p_1(X) = ||M_1^{-1}|_{m_1}(x_1 - x_{23})|_{m_1} \tag{3}$$

The only comparator for moduli set τ is proposed in [20], which compares signed numbers. This comparator is based on an optimized version of MRC for sign identification and performs the comparison through utilizing the sign bits of the operands as well as their difference. The authors of [20] assumed that the inputs of the comparison method are two RNS numbers with two extra bits that identify their signs. Such a comparator can be used in an architecture that, after performing any RNS operations, the sign of the result is detected and stored. For multiplication/division operations, the sign of the result is identified with a simple XOR gate in parallel with the multiplication/division computation and does not cause any overhead. Unlike multiplication/division, the sign of the addition/subtraction result is identified by comparing the result with $[M/2]$ (via MRC digits). Therefore, the adder and subtractor should be augmented with the comparator.
To capture the overhead cost of the sign-detection circuit after each RNS operation, we have synthesized a modular addition with or without the sign identification circuit regarding moduli set τ for $n = 8$ and $k = 0$ via TSMC 90-nm standard CMOS technology by Synopsys Design Compiler for delay constraints in a range of $0.7 - 2.0ns$ with $0.1 - ns$ intervals. The power measures are illustrated by the curves in Figure 1.

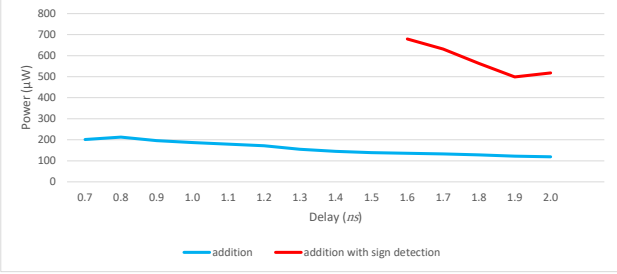

Figure 1. Comparison of power dissipation for simple addition and addition with sign detection

The delay of the addition in moduli set τ with and without the sign-detection circuit is 1.6 and $0.7ns$, respectively. Augmenting the sign-detection circuit after each addition/subtraction causes up to twice the delay and power dissipation than simple addition/subtraction. Such a growth in the power and delay is in contrast with the strength points of RNS, which makes it an appropriate candidate for applications with repeated usage of addition and multiplication operations.

3. Proposed DRP-based signed integer comparator

The proposed comparator is presented as Algorithm 1. First, the signs of the operands are identified; then, if their signs are identical, the comparison operation is performed by comparing their corresponding DRP components. The E, G, and L values in this function denote the equality of the inputs, $X > Y$, and $X < Y$, respectively. In the following, the proposed sign-detection method is first explained; it is based on the DRP method [16]. Then, the computation of the DRP components in moduli set τ are designated.

Algorithm 1: Signed number comparison in moduli set τ

1. function comparison (inputs: $X : (x_1, x_2, x_3)$,$Y : (y_1, y_2, y_3)$, output: $Comp$)
2. if $sign(X) = sign(Y)$
3. if $(p_1(X) > p_1(Y))$ then $Comp = G$ // $X > Y$
4. else if $(p_1(X) < p_1(Y))$ then $Comp = L$ // $X < Y$
5. else if $(p_2(X) > p_2(Y))$ then $Comp = G$ // $X > Y$
6. else if $(p_2(X) < p_2(Y))$ then $Comp = L$ // $X < Y$
7. else $Comp = E$ // $X = Y$
8. else if $sign(X) = 0$ then $Comp = G$ // $X > Y$
9. else $Comp = L$ // $X < Y$
10. return $Comp$
11. end function

3.1. Sign detection

Let the $m_1 = 2^{n+k}$, $m_2 = 2^n - 1$, and $m_3 = 2^n + 1$ DRP method described in (2) and (3) splits the dynamic range into 2^{n+k} equal partitions in the moduli set τ (each of which contains consecutive numbers – see Table 1). Each number in the dynamic range belongs to exactly one partition. In each row of Table 1, a partition of the dynamic range with its primary and secondary integer identifiers (i.e., $p_1(X)$ and $p_2(X)$) are presented, where $M_1 = M/m_1 = 2^{2n} - 1$. As shown in Table 1, the values of $p_2(X)$ are different in each subset of the dynamic range that has a unique $p_1(X)$. For example, the values of $p_2(X)$ for the first row of Table 1 with $p_1(X) = 0$ are presented in Table 2.

In moduli set τ, $[0, 2^{n+k-1}(2^{2n} - 1))$ and $[2^{n+k-1}(2^{2n} - 1), 2^{n+k}(2^{2n} - 1))$ are the intervals for the positive and negative numbers, respectively. As was proven in Theorem 1, the sign of an RNS number $X = (x_1, x_2, x_3)$ is identified by the value of

$p_1(X)$. Theorem 1: An RNS number X in moduli set τ is negative if and only if $p_{1,n+k-1} = 1$, where $p_1(X) = p_{1,n+k-1} \cdots p_{1,0}$.
Proof: $p_{1,n+k-1} = 1$ indicates that $p_1(X) \geq 2^{n+k-1}$. Equation (4) describes X in terms of DRP components, where $p_1(X) \in [0, 2^{n+k})$, $M_1 = 2^{2n} - 1$, and $x_{23} = |X|_{2^{2n}-1} \in [0, 2^{2n} - 1)$.

$$X = p_1(X)M_1 + x_{23} = p_1(X)(2^{2n} - 1) + x_{23} \tag{4}$$

Based on (4) and assuming $p_1(X) < 2^{n+k-1}$, this entails $X < 2^{n+k-1}(2^{2n} - 1) + x_{23}$, which shows that $X > 0$. With similar explanations, it is clear that $p_1(X) \geq 2^{n+k-1}$ leads to $X \geq 2^{n+k-1}(2^{2n} - 1)$, which indicates that X is negative.

Table 1
$p_1(X)$ values for moduli set τ

subset of dynamic range	$p_1(X)$	$p_2(X)$
$[0, M_1)$	0	$[0, 2^n - 1)$
$[M_1, 2M_1)$	1	$[0, 2^n - 1)$
...	...	...
$[2^n M_1, (2^n + 1)M_1)$	2^n	$[0, 2^n - 1)$
...	...	...
$[(2^{n+k} - 1)M_1, 2^{n+k}M_1)$	$2^{n+k} - 1$	$[0, 2^n - 1)$

Table 2
values of $p_1(X)$ and $p_2(X)$ for $0 \leq X < (2^{2n} - 1)$

subset of dynamic range	$p_1(X)$	$p_2(X)$
$[0, m_3)$	0	0
$[m_3, 2m_3)$	0	1
...	...	...
$[(2^n - 2)m_3, (2^n - 1)m_3)$	0	$2^n - 2$

3.2. Computation of DRP components

$p_1(X)$ and $p_2(X)$ are investigated for moduli set τ. The multiplicative inverses that are required for the DRP components are represented via Properties 1 and 2. Based on (2), (3), and Properties 1 and 2, $p_1(X)$ and $p_2(X)$ are described in (5) and (6).

Property 1: $|(2^n + 1)^{-1}|_{2^n-1} = 2^{n-1}$

Property 2: $|(2^{2n} - 1)^{-1}|_{2^{n+k}} = -1$

$$p_1(X) = ||(2^{2n} - 1)^{-1}|_{2^{n+k}}(x_1 - x_{23})|_{2^{n+k}} = |x_{23} - x_1|_{2^{n+k}} \tag{5}$$

$$p_2(X) = ||(2^n + 1)^{-1})|_{2^n-1}(x_2 - x_3)|_{2^n-1} = |2^{n-1}(x_2 - x_3)|_{2^n-1} \tag{6}$$

4. Implementation

The actual delay and cost of the proposed comparison method directly depend on the complexity of the $p_1(X)$ and $p_2(X)$ generators. In this section, we provide implementation details for generating $p_1(X)$ and $p_2(X)$. To drive the implementation-friendly equations for the DRP components (in consideration of $\overline{x_3} = 2^{n+1}-1-x_3$), $p_2(X)$ can be simplified to (7). Given that $|2^{n-1}x_2 + 2^{n-1}\overline{x_3} - 2^{n-1}|_{2^n-1} = |U+V|_{2^n-1}$, U and V are obtained through a modulo(2^n-1) carry save adder (CSA), where $U+V=w$ and $w = w_n w_{n-1} \cdots w_0$. Equation (8) is obtained by the well-known property of the modulo 2^n-1 arithmetic (i.e., $|2^n w_n|_{2^n-1} = w_n$). Table 3 illustrates weighted-bit arrangements for the three terms of $p_2(X)$, where $U = u_{n-1} \cdots u_0$, $V = v_{n-1} \cdots v_0$ and $b_{n-1} \cdots b_0$, $c_n \cdots c_0$ represent x_2 and x_3, respectively.

$$p_2(X) = |2^{n-1}x_2 + 2^{n-1}(\overline{x_3} - 2^{n+1} + 1)|_{2^n-1}$$

$$= |2^{n-1}x_2 + 2^{n-1}\overline{x_3} - 2^{n-1}|_{2^n-1} \tag{7}$$

$$= |U+V|_{2^n-1} = w - 2^n w_n + w_n \tag{8}$$

Table 3
Bit organization of $p_2(X)$

2^{n-1}	...	2^1	2^0	Component
b_0	...	b_2	b_1	$\lvert 2^{n-1}x_2\rvert_{2^n-1}$
$\overline{c_0}$	...	$\overline{c_2}$	$\overline{c_1}$	$\lvert 2^{n-1}\overline{x_3}\rvert_{2^n-1}$
$\overline{c_n}$	...	1	1	$\lvert -2^{n-1}\rvert_{2^n-1}$
u_{n-1}	...	u_1	u_0	U
v_{n-1}	...	v_1	v_0	V

By the use of a new CRT [26] and (8), we replace x_{23} with $x_3+(2^n+1)|U+V|_{2^n-1}$ in (5), which leads to (9), where $\overline{x_1} = 2^{n+k} - 1 - x_1$. The bit organization of $p_1(X)$ is illustrated in Table 4, where $a_{n+k-1} \cdots a_0$ represents x_1.

$$p_1(X) = |x_3 + (2^n+1)|2^{n-1}(x_2 - x_3)|_{2^n-1} - x_1|_{2^{n+k}}$$

$$= |x_3 + (2^n+1)|U+V|_{2^n-1} - x_1|_{2^{n+k}}$$

$$= |x_3 + (2^n+1)(w - 2^n w_n + w_n) - x_1|_{2^{n+k}}$$

$$= |x_3 + 2^n w + w + w_n - x_1|_{2^{n+k}}$$

$$= |x_3 + (2^n+1)w + w_n + \overline{x_1} + 1|_{2^{n+k}}$$

$$= |x_3 + (2^n+1)(U+V) + w_n + \overline{x_1} + 1|_{2^{n+k}} \tag{9}$$

Table 4
Bit organization of $p_1(X)$

2^{n+k-1}	...	2^n	2^{n-1}	2^{n-2}	....	2^0
		c_n	c_{n-1}	c_{n-2}	...	c_0
u_{k-1}	...	u_0	u_{n-1}	u_{n-2}	...	u_0
v_{k-1}	...	v_0	v_{n-1}	v_{n-2}	...	v_0
$\overline{a_{n+k-1}}$	...	$\overline{a_n}$	$\overline{a_{n-1}}$	$\overline{a_{n-2}}$	...	$\overline{a_0}$
						1
						w_n

The value of $p_2(X)$ is achieved after the n-bit modulo (2^n-1) CSA, whose output (i.e., U and V) feed an n-bit modular adder. The implementation of $p_1(X)$ includes an array of $_2^4C(4:2 compressor)$ to reduce the number of operands that exclude w_n to two. Two adders regarding the two possible values of w_n received the outputs of $_2^4C$ (as shown in Figure 2). To obtain $p_1(X)$, the output of these two adders is multiplexed through w_n. In another realization ($New_{Red.}$), two $(n+k)$-bit adders are reduced to one $(n+k)$-bit adder – the carry of which is w_n, where *Red.* indicates the reduced version of the proposed architecture.

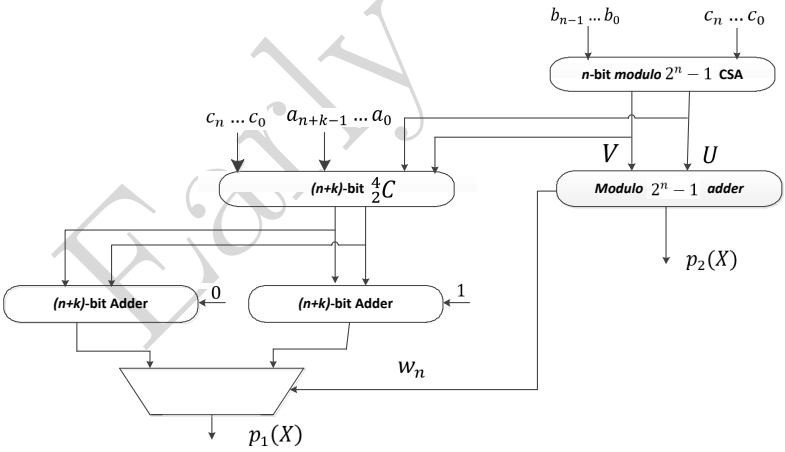

Figure 2. Proposed architecture of $p_1(X)$ and $p_2(X)$ generators for the moduli set τ

4.1. Parallel prefix realization

In the high-speed design of the proposed comparator (called New_{PPA}), the two simple $(n+k)$-bit adders of Figure 2 can be replaced by one parallel prefix adder (PPA) [11], which includes a carry input (as shown in Figure 3). In the PPA shown in Figure 2, the carry generation signal g_i and carry propagation signal p_i for each position $i \in (0, n+k]$

for two inputs $u = u_{n+k-1} \cdots u_0$ and $v = v_{n+k-1} \cdots v_0$ are computed as follows:

$$g_i = u_i \wedge v_i, p_i = u_i \vee v_i$$

Using g_i and p_i, group propagate $P_{i:j}$ and group generate $G_{i:j}$ are computed by indicating the carry generation and propagation ability within positions j to i, where $j < i$, $G_{i:0} = G_i$, and $P_{i:0} = P_i$. After the computation of carries c_i, sum bits s_i can be computed in a straightforward way as

$$h_i = u_i \oplus v_i, s_i = h_i \oplus c_i.$$

On the other hand, the prefix design of PPA can be extended to achieve the sum of two operands and a carry input (i.e., w_n) (as shown in Figure 3). Therefore, the critical delay path (CDP) of this design consists of an array of CSA, an n-bit modular adder, one-stage PPA, and an XOR array. Figure 3 depicts the required PPA for generating $p_1(X)$ in the aforementioned design.

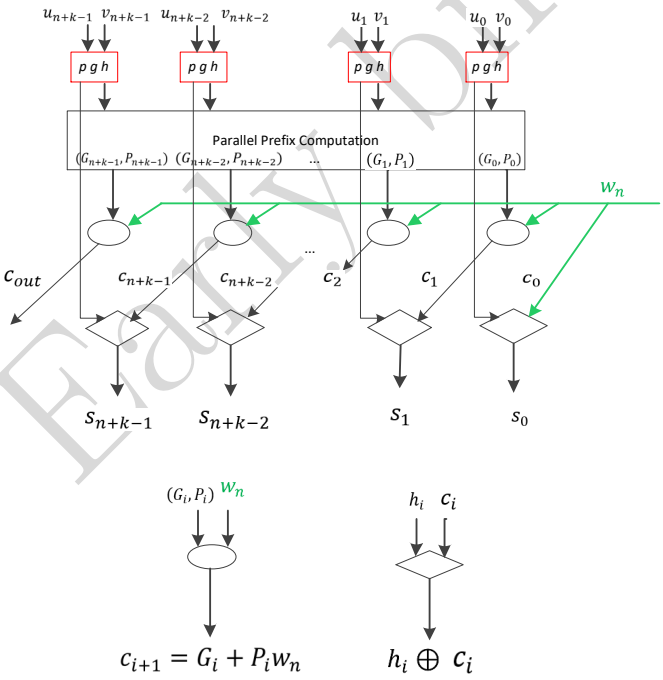

Figure 3. PPA architecture of adder in New_{PPA}, which includes carry as input (i.e., w_n)

An evaluation of the proposed designs is discussed in the next section. If the operands have different signs, a final comparison of the two operands ($X = (x_1, x_2, x_3)$ and $Y = (y_1, y_2, y_3)$) is accomplished through feeding set of operands $[p_1(X), p_1(Y)]$,

$[p_2(X), p_2(Y)]$, $[x_3, y_3]$ into three different comparators. In other words, the final comparison operation can be reduced to primarily comparing $p_1(X)$ and $p_1(Y)$. In the event of $p_1(X) = p_1(Y)$, a comparison of $p_2(X)$ and $p_2(Y)$ can lead to the overall comparison result unless they are also equal; in such a case, a comparison of the modulo-$(2^n + 1)$ residues yields the final result.

5. Discussion

The proposed comparator method in this paper includes $p_1(X)$ and $p_2(X)$ generators followed by three comparators. Based on the $p_1(X)$ generator, three different implementations (namely, New, $New_{Red.}$, and New_{PPA}) are proposed. New is exactly based on Figure 2, while $New_{Red.}$ and New_{PPA} eliminate the multiplexer that selects the desired $p_2(x)$. In addition, the two $(n + k)$-bit adders compound into an $(n+k)$-bit PPA and an $(n+k)$-bit carry ripple adder (CRA) to save costs in $New_{Red.}$ and New_{PPA}, respectively.

As discussed in Section 2, the comparator of [20] is applicable to the architectures that have a sign-detection circuit after each operation (which leads to extra power consumption). Therefore, we assume that, in the comparator of [20], the signs of the operands are computed like their difference, which leads to triple the same hardware or execution time. Here, we compare three different proposed schemas with the straightforward comparator and the comparator of [20]. The straightforward comparator includes the best RNS reverse converter [8] for moduli set τ and a normal binary comparator.

In Tables 5 and 6, the number of different components within the CDP and all of the components in each design are illustrated, respectively. Based on the components of each design in Table 5, the comparators of [8, 20] have more components within the CDP, which leads to more delay than the proposed architectures. Indeed, the comparators of [8,20] have three different adders, while New and New_{PPA} have only two adders within the CDP.

Table 5
Number of components within CDP

Method	CSA	CRA			PPA
		n-bit	$n + k$-bit	$3n + k$-bit	$n + k$-bit
New	3		2		
$New_{Red.}$	1	1	2		
New_{PPA}	2	1	1		
[20]	3		2		1
[8]	1	2		1	

The total delays and costs of the proposed architectures as well as those of [8,20] are described in Table 7. The maximum and minimum total cost of the τ-comparators belong to the comparators of [20] and [8], respectively. The comparator of [8] has less

cost than the proposed comparators; however, its delay is drastically greater than the others. The delay comparison is based on gate counting within the CDP, where the delay of a simple two-input gate (i.e., AND, OR, NAND, and NOR) is denoted by $\triangle G$ [11]. An n-bit CRA or CSA have a $2n \triangle G$ delay and $7n\#G$ cost, while an n-bit PPA has a $(2log n + 3) \triangle G$ delay and $(1.5nlog(n+1) + 5n)\#G$ cost.

Table 6
Number of components in different comparators

Method	Simple Gate	CSA			CRA			PPA	
		k-bit	n-bit	$n+k$-bit	n-bit	$n+k$-bit	$3n+k$-bit	n-bit	$n+k$-bit
New	$log(3n+k)$		2	4	4	5			
$New_{Red.}$	$log(3n+k)$		2	3	4	5			
New_{PPA}	$log(3n+k)$		2	3	2	5		2	
[20]	$3log(3n+k)$	3		3	6	6		3	3
[8]	$12k+2n$		4		4		1		

Table 7
Total delay and cost in RNS comparators

Method	Total delay ($\triangle G$)	Total cost ($\#G$)
New	$4n+4k+12$	$95n+63k+log(3n+k)$
$New_{Red.}$	$6n+4k+4$	$88n+56k+log(3n+k)$
New_{PPA}	$4n+2k+8$	$91n+63k+3nlogn+log(3n+k)$
[20]	$4(n+k)+2log(n+k)+15$	$135n+99k+log(3n+k)+4.5n(log(n)+log(n+k))$
[8]	$10n+2k+4$	$79n+19k$

In order to find better insights into the merits of the proposed architectures, the total delay and cost of each design are compared via the plots of Figures 4 and 5 for $n = 8$, 16, where $k \leq n$. As shown in Figures 4 and 5, the cost of the comparator of [20] and the delay of [8] are each more than twice as much as those in the proposed architectures. Based on the plots in Figure 4, the delay of the two proposed architectures (New and New_{PPA}) are less than the comparators of [8] and [20]. In addition, this confirms the superiority of the New_{PPA} in terms of delay with increased k. In conclusion, the best previous τ-comparator [20] has more delay and cost as compared to the proposed method. The lower cost of the straightforward comparator [8] is achieved at a cost of much more delay. The considerably high delay of [8] (which is in contrast to the RNS properties) makes it inefficient for RNS applications. The performances of the proposed architectures are better than their counterparts; among the three proposed architectures, $New_{Red.}$ and New_{PPA} have less cost and less delay, respectively.

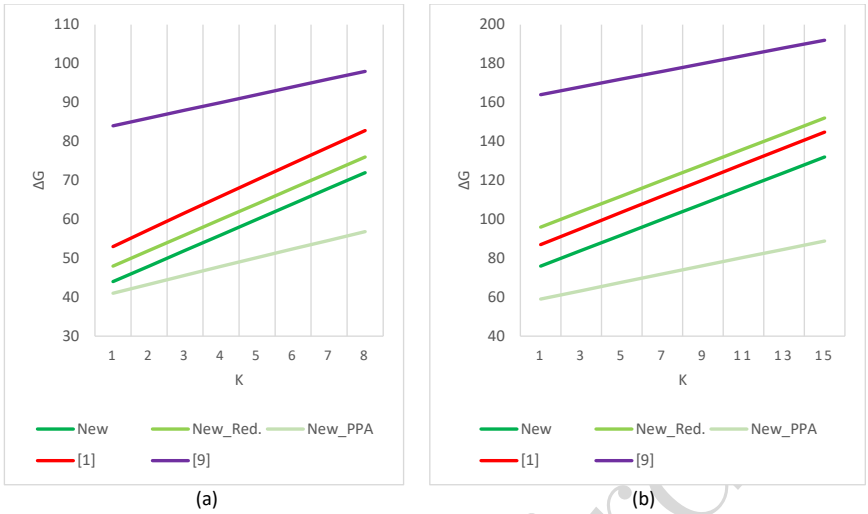

Figure 4. Delays of New, New_{PPA}, $New_{Red.}$, and two previous τ-comparators for a) $n = 8$ and b) $n = 16$

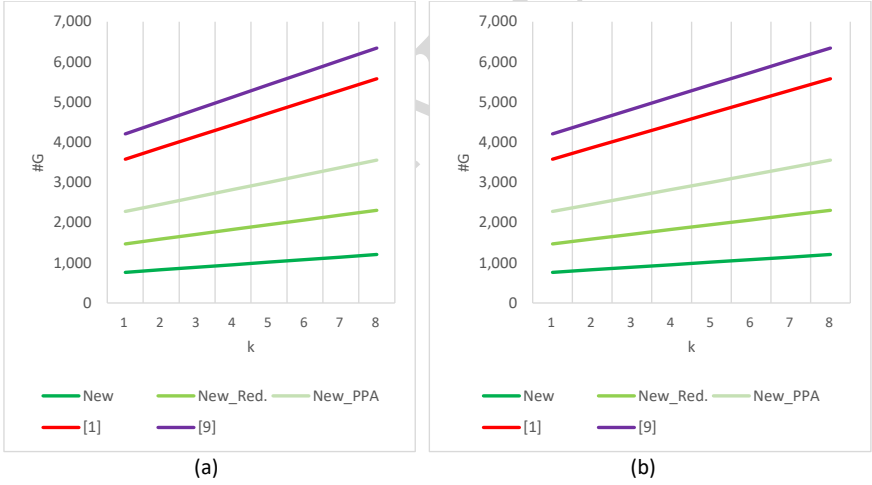

Figure 5. Costs of New, New_{PPA}, $New_{Red.}$, and two previous τ-comparators for a) $n = 8$ and b) $n = 16$

6. Conclusion

Because of the costly and time-consuming comparison operation in a residue number system, the most common applications that use this system feature comparison-free

computations. However, some research efforts have been ongoing in order to realize efficient comparators to widen the application of this number system. Moduli set τ is an extension of popular moduli set $\{2^n - 1, 2^n, 2^n + 1\}$, whose reverse converter has been appeared in the relevant literature.

Dynamic range partitioning with partitioning the dynamic range to several intervals helps to facilitate complicated operations. With the use of DRP components, the proposed comparator identified the sign of the operands and compared them. An evaluation of the proposed designs and the best previous comparator showed that the proposed reduced design has less delay and that the proposed parallel prefix design has less cost for $n = 8$ and $n = 16$.

References

[1] Alhassan I.Z., Ansong E.D., Abdul-Salaam G., Alhassan S.: Enhancing Image Security during Transmission using Residue Number System and k-shuffle, *Earthline Journal of Mathematical Sciences*, vol. 4(2), pp. 399–424, 2020.

[2] Bi S., Gross W.J.: The mixed-radix Chinese remainder theorem and its applications to residue comparison, *IEEE Transactions on Computers*, vol. 57(12), pp. 1624–1632, 2008.

[3] Boyvalenkov P., Chervyakov N.I., Lyakhov P., Semyonova N., Nazarov A., Valueva M., Boyvalenkov G., Bogaevskiy D., Kaplun D.: Classification of Moduli Sets for Residue Number System With Special Diagonal Functions, *IEEE Access*, vol. 8, pp. 156104–156116, 2020.

[4] carlo Cardarilli G., Di Nunzio L., Fazzolari R., Nannarelli A., Petricca M., Re M.: Design Space Exploration based Methodology for Residue Number System Digital Filters Implementation, *IEEE Transactions on Emerging Topics in Computing*, 2020.

[5] Dimauro G., Impedovo S., Pirlo G., Salzo A.: RNS architectures for the implementation of thediagonal function', *Information processing letters*, vol. 73(5-6), pp. 189–198, 2000.

[6] Gbolagade K., Chaves R., Sousa L., Cotofana S.: Residue-to-binary converters for the moduli set {2 2n+ 1-1, 2 2n, 2 n-1}. In: *2009 2nd International Conference on Adaptive Science & Technology (ICAST)*, pp. 26–33, IEEE, 2009.

[7] Gbolagade K.A., Chaves R., Sousa L., Cotofana S.D.: An improved RNS reverse converter for the {2 2n+ 1- 1, 2 n, 2 n- 1} moduli set. In: *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pp. 2103–2106, IEEE, 2010.

[8] Hiasat A.: A residue-to-binary converter with an adjustable structure for an extended RNS three-moduli set, *Journal of Circuits, Systems and Computers*, vol. 28(08), p. 1950126, 2019.

[9] Hiasat A., Sousa L.: Sign Identifier for the Enhanced Three Moduli Set $\{2^{n+k}, 2^n - 1, 2^{n+1} - 1\}$, *Journal of Signal Processing Systems*, vol. 91(8), pp. 953–961, 2019.

[10] Isupov K.: Using Floating-Point Intervals for Non-Modular Computations in Residue Number System, *IEEE Access*, vol. 8, pp. 58603–58619, 2020.

[11] Kalampoukas L., Efstathiou C., Nikoloo D., Vergos H.T., Kalamatianos J.: High--speed parallel-prefix modulo 2n-1 adders, 2006. US Patent 7,155,473.

[12] Krasnobayev V., Yanko A., Koshman S.: A Method for arithmetic comparison of data represented in a residue number system, *Cybernetics and Systems Analysis*, vol. 52(1), pp. 145–150, 2016.

[13] Kumar S., Chang C.H., Tay T.F.: New algorithm for signed integer comparison in $2^{n+k}, 2^n - 1, 2^n + 1, 2^{n\pm1} - 1$ and its efficient hardware implementation, *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 64(6), pp. 1481–1493, 2016.

[14] Latha M.M., Rachh R.R., Mohan P.A.: RNS-to-Binary Converters for a Three-Moduli Set {2 n- 1- 1, 2 n- 1, 2 n+ k}, *IETE Journal of Education*, vol. 58(1), pp. 20–28, 2017.

[15] Mohan P.V.A.: RNS-To-Binary Converter for a New Three-Moduli Set $2^{n+1} - 1, 2^n, 2^n - 1$, *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 54(9), pp. 775–779, 2007.

[16] Salamat S., Imani M., Gupta S., Rosing T.: Rnsnet: In-memory neural network acceleration using residue number system. In: *2018 IEEE International Conference on Rebooting Computing (ICRC)*, pp. 1–12, IEEE, 2018.

[17] Samimi N., Kamal M., Afzali-Kusha A., Pedram M.: Res-DNN: A Residue Number System-Based DNN Accelerator Unit, *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67(2), pp. 658–671, 2019.

[18] Schoinianakis D.: Residue arithmetic systems in cryptography: a survey on modern security applications, *Journal of Cryptographic Engineering*, vol. 10(3), pp. 249–267, 2020.

[19] Sousa L.: Efficient method for magnitude comparison in RNS based on two pairs of conjugate moduli. In: *18th IEEE Symposium on Computer Arithmetic (ARITH'07)*, pp. 240–250, IEEE, 2007.

[20] Sousa L., Martins P.: Sign Detection and Number Comparison on RNS 3-Moduli Sets $\{2^n - 1, 2^{n+k}, 2^n + 1\}$, *Circuits, Systems, and Signal Processing*, vol. 36(3), pp. 1224–1246, 2017.

[21] Szabo N.S., Tanaka R.I.: *Residue arithmetic and its applications to computer technology*, McGraw-Hill, 1967.

[22] Torabi Z., Jaberipur G.: Low-power/cost RNS comparison via partitioning the dynamic range, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24(5), pp. 1849–1857, 2015.

[23] Torabi Z., Jaberipur G.: Low-power/cost RNS comparison via partitioning the dynamic range, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24(5), pp. 1849–1857, 2015.

[24] Torabi Z., Jaberipur G.: Fast low energy RNS comparators for 4-moduli sets $\{2^{n\pm1}, 2^n, m\}$ with $\in \{2^{n+1} \pm 1, 2^{n-1} - 1\}$, *Integration*, vol. 55, pp. 155–161, 2016.

[25] Torabi Z., Jaberipur G., Belghadr A.: Fast division in the residue number system {2n+ 1, 2n, 2n-1} based on shortcut mixed radix conversion, *Computers & Electrical Engineering*, vol. 83, p. 106571, 2020.

[26] Wang Y.: New Chinese remainder theorems. In: *Conference Record of Thirty-Second Asilomar Conference on Signals, Systems and Computers (Cat. No. 98CH36284)*, vol. 1, pp. 165–171, IEEE, 1998.

[27] Wang Y., Song X., Aboulhamid M.: A new algorithm for RNS magnitude comparison based on new Chinese remainder theorem II. In: *Proceedings Ninth Great Lakes Symposium on VLSI*, pp. 362–365, IEEE, 1999.

[28] Youssef M., Emam A.E., Abd Elghany M.: Image multiplexing using residue number system coding over MIMO-OFDM communication system., *International Journal of Electrical & Computer Engineering (2088-8708)*, vol. 9, 2019.

Affiliations

Zeinab Torabi
Shahid Rajaee Teacher Training University, Department of Computer Engineering, Tehran, Iran, z.torabi@sru.ac.ir

Somayeh Timarchi
Shahid Beheshti University, Faculty of Electrical Engineering, Tehran, Iran, s_timarchi@sbu.ac.ir

Received: ??.??.2021
Revised: ??.??.2021
Accepted: ??.??.2021